1 Objectives

In order to express the meaning of a program, we need a formal language to capture these meanings. One way to express meaning is to say something about the *types* of the expressions. By the end of lecture, you should know

- what the word "semantics" means.
- how to structure a proof-tree
- how use the type rules to prove the type of an expression
- write your own type rule for an expression

2 Type Rules

Arithmetic $$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 \oplus e_2 : \texttt{int}}$$

Relations $$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 \sim e_2 : \texttt{bool}}$$

Booleans $$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1 \texttt{\&\&}\ e_2 : \texttt{bool}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{bool}}{\Gamma \vdash e_1 \texttt{||}\ e_2 : \texttt{bool}}$$

If $$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 : \tau}$$

Application $$\frac{\Gamma \vdash e : \tau_1 \to \tau_2 \to \cdots \to \tau_n \to \tau \quad \Gamma \vdash e_1 : \tau_1 \quad \cdots \quad \Gamma \vdash e_n : \tau_n}{\Gamma \vdash e\ e_1\ e_2\ \cdots\ e_n : \tau}$$

Functions $$\frac{\Gamma \cup [\texttt{x}_1 : \tau_1; \cdots; \texttt{x}_n : \tau_n] \vdash e : \tau}{\Gamma \vdash \texttt{fun}\ \texttt{x}_1 \cdots \texttt{x}_n \texttt{->} e : \tau_1 \to \cdots \to \tau_n \to \tau}$$

Let $$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \cup [x : \tau] \vdash e_2 : \tau'}{\Gamma \vdash \texttt{let}\ \texttt{x} = e_1\ \texttt{in}\ e_2 : \tau'}$$

$$\frac{\Gamma \cup [x : \tau] \vdash e_1 : \tau \quad \Gamma \cup [x : \tau] \vdash e_2 : \tau'}{\Gamma \vdash \texttt{let rec}\ \texttt{x} = e_1\ \texttt{in}\ e_2 : \tau'}$$

3 Problems

Try these problems. In a few minutes the instructor will go over the solutions. Feel free to work with the person next to you!

- Prove that $\Gamma \vdash$ `let` $f =$ `fun` x`->`x`+2` `in` `let` $g =$ `fun` x`->`x`+3` `in` (`if` $4 > 6$ `then` f `else` g) 10 : `int`
- Write a type judgment rule for the list operator `::`.